AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (original) A method of enabling multiple different operating systems to run concurrently on the same computer, comprising: selecting a first operating system to have a relatively high priority; selecting at least one second operating system to have a relatively lower priority; providing a common program arranged to switch between said operating systems under predetermined conditions ; and providing modifications to said first and second operating systems to allow them to be controlled by said common program.

2. (original) The method of claim 1, wherein the first and second operating systems are associated with first and second memory contexts, respectively, and the common program is associated with a third memory context, the method comprising switching a current memory context to the first, second or third memory context when switching between said operating systems.

3. (original) The method of claim 2, further comprising switching the current memory context to the first memory context when switching to or from the first operating system.

1057858

4. (currently amended) The method of claim 2 or 3, further comprising invoking the common program by the first operating system, and starting execution of the common program in the first memory context.

5. (currently amended) The method of claim 2 or 3, further comprising preempting the first operating system by the common program, and starting execution of the common program in the first memory context.

6. (original) The method of claim 2, comprising: switching the current memory context to the third memory context when switching from the second operating system.

7. (original) The method of claim 6, further comprising invoking the common program by the second operating system, wherein the current memory context is the third memory context.

8. (original) The method of claim 6, further comprising preempting the second operating system by the common program, wherein the current memory context is the third memory context.

9. (original) The method of claim 8, wherein the second operating system invokes the common program by a trap call.

10. (currently amended) The method of ~~any preceding~~ claim 1, in which the first operating system is a real time operating system.

11. (currently amended) The method of ~~any preceding~~ claim 1, in which the second operating system is a non-real time, general-purpose operating system.

12. (currently amended) The method of ~~any preceding~~ claim 1, in which the second operating system is Linux, or a version or variant thereof.

13. (currently amended) The method of ~~any preceding~~ claim 1, in which the common program is arranged to save, and to restore from a saved version, the processor state required to switch between the operating systems.

14. (currently amended) The method of ~~any preceding~~ claim 1, in which processor exceptions for the second operating system are handled in virtual fashion by the common program.

1057858

15. (currently amended) The method of ~~any preceding~~ claim 1, in which the common

program is arranged to intercept some processor exceptions, and to call exception

handling routines of the first operating system to service them.

16. (currently amended) The method of ~~any preceding~~ claim 1, in which the processor

exceptions for the second operating system are notified as virtual exceptions.

17. (original) The method of claim 16, in which the common program is arranged to call

an exception handling routine of the second operating system corresponding to a said

virtual exception which is pending.

18. (currently amended) The method of ~~any preceding~~ claim 1, further comprising

providing each of said operating systems with separate memory spaces in which each

can exclusively operate.

19. (currently amended) The method of ~~any preceding~~ claim 1, further comprising

providing each of said operating systems with first input and/or output devices of said

computer to which each has exclusive access.

1057858

20. (currently amended) The method of ~~any preceding~~ claim 1, in which each operating

system accesses said first input and/or output devices using substantially modified

native routines.

21. (currently amended) The method of ~~any preceding~~ claim 1, further comprising

providing each of said operating systems with access to second input and/or output

devices of said computer to which each has shared access.

22. (original) The method of claim 21, in which all operating systems access said

second input and/or output devices using the routines of the first operating system.

23. (currently amended) The method of ~~any preceding~~ claim 1, further comprising

providing a restart routine for restarting a said second operating systems without

interrupting operation of said first, or said common program.

24. (original) The method of claim 21, in which said second device comprises a co-

processor, and in which, on switching between said first operating system and said

second (or vice versa), the state of said co-processor is not changed, whereby if said

1057858

operating systems switch back without intervening access to said coprocessor, its

operation can complete uninterrupted.

25. (original) The method of claim 1, in which one or more original address tables are

provided by the computer for use by an operating system, and in which the common

program accesses said original address tables, and provides a plurality of replicated

tables having the same structure as said original tables, elsewhere in memory, one per

table per operating system, each for use by a respective operating system, and in which

said operating systems are modified so as to replace instructions which write said

original address tables with routine calls which access said replicated tables.

26. (currently amended) The method of any preceding-claim 1, further comprising

combining said operating systems and common program into a single code product.

27. (currently amended) The method of any preceding-claim 1, further comprising

embedding said operating systems and common program onto persistent memory on a

computer product.

28. (currently amended) The method of any preceding claim 1, in which each said

operating system is provided with an idle routine, in which it passes control to the

common program.

29. (original) The method of claim 28, in which said idle routine substitutes for a

processor halt instruction.

30. (currently amended) A development kit computer program product comprising code

for performing the steps of any preceding claim 1.

31. (original) A computer program product comprising code combined according to

claim 30.

32. (original) An embedded computer system comprising a CPU, memory devices and

input/output devices, having stored on persistent memory therein programs embedded

according to claim 31.

33. (original) A computer system comprising a CPU, memory devices and input/output

devices, having executing thereon computer code comprising; a first operating system

having a relatively high priority; a second operating system having a relatively lower

priority; and a common program arranged to run said operating systems concurrently by

switching between said operating systems under predetermined conditions.

34. (currently amended) A computer system according to claim 33, arranged to run said

first and second operating systems concurrently using the method ~~of any of claims 1 to~~

~~29~~ as described above.

35. (original) A computer system comprising a processor and a memory and operable to

execute thereon computer code to operate first and second operating systems in first

and second memory contexts, respectively, and a common program operable in said

first or a third memory context to switch between the first and second operating

systems, wherein the memory context in which the common program is operated

depends on the switching operation.

36. (currently amended) The system, product or method of ~~any preceding~~ claim 1 in

which the computer has a Complex Instruction Set architecture.